Proceedings of the 1st CIRP Industrial Product-Service Systems (IPS2) Conference, 1-2 April 2009, pp54

A Method of Supporting Conflict Resolution for Designing Services

Y. Akiyama[1], Y. Shimomura[1], T. Arai[2]

[1]Department of System Design, Tokyo Metropolitan University, Asahigaoka 6-6, Hino-shi, Tokyo, 191-0065, Japan

[2]Department of Precision Engineering, The University of Tokyo, Hongo 7-3-1, Bunkyo-ku, Tokyo, 113-8656, Japan

akiyama-yoshiki@sd.tmu.ac.jp, yoshiki-shimomura@center.tmu.ac.jp, arai-tamio@robot.t.u-tokyo.ac.jp

Abstract

Recently, the importance of service has been emphasized in various industries. However, few studies have focused on service design in spite of its great relevance. This paper proposes a methodology for service design, which enables designers to determine existing conflicts in design solutions and obtain basic strategies to solve them using computers. Two different approaches for detecting conflicts are proposed; one is the use of lexical expressions of functions, and the other involves the ranges of design parameters. At the end, the verification of the proposed methodology is carried out through application to an existing case.

Keywords:
Service Engineering, Design Support, CAD

1 INTRODUCTION

Because the material needs in society are largely met, consumers regard the total value from a package of products and services as more important than that from products alone. Especially, in manufacturing, service activities previously considered only as additional elements, such as maintenance, repair, installation, consulting, and lease, are regarded as essential elements to make a business more competitive. That is to say, products and services are inextricably linked, and finding the methodology and tools to design products and services on a common platform is imperative. However, in spite of the importance of service, few studies have forcused on service design or its evaluation (e.g., [1, 2, 3, 4, 5]). Under these circumstances, studies on Service Engineering (SE) (e.g., [6, 7]) have focused on the effective integration of products and services. SE provides a methodology from the viewpoint of engineering and makes it possible for designers to develop design solutions without relying on trial and error. Researchers on SE have also been developing an integrated design-support tool for products and services, called Service CAD system [8].

In SE, studies on the description of a design solution and its elaboration (e.g., [2]) have mainly been conducted. On the other hand, design is viewed as problem solving to satisfy customer demands. However, an insufficient number of studies on the method that supports this process have been carried out. In service design, an ad hoc design process with trial and error is still dominant, and the quality of the design solution largely depends on a designer's experience and intuition. The authors consider the conflicts in design solution to be a key to the creation of quality design. Generally, design solutions obtained with the current Service CAD frequently present various conflicts, such as those that make it difficult to implement a design solution. In other words, the quality and efficiency of design are largely dependent on how rapidly designers can discover and solve such conflicts.

In this paper, a methodology for supporting design is proposed which enables designers to determine the existing conflicts in design solutions and obtain the basic strategies to solve them using computers. Though many types of conflicts exist (e.g. conflict of requirements between a service provider and consumers), only conflict in parameter values in a service design solution described in conceptual design stage is dealt with. Since a service is described as a set of various kinds of parameters and their state transitions in SE methodology, conflicts can be detected without additional information. Two approaches for detecting conflicts are proposed; one is the use of the lexical expressions of functions, and the other involves the ranges of design parameters. Methods for solving the detected conflict with TRIZ [9] methodology are also suggested. The use of TRIZ methodology helps service designers solve problems without relying on trial and error. In addition, the methods are verified by application to an existing case.

2 EXISTING STUDIES ON CONFLICT-SOLVING

TRIZ is a well-known methodology that provides for the detection and resolution of conflicts in the product design field. TRIZ consists of various methods and a knowledge base grounded on former inventions that enable designers to effectively solve problems. For example, the Technical Contradiction Table [9, 10] suggests the principles to solve conflicts between elements in a design solution. The principles are created on the basis of the analysis of millions of patents. The use of TRIZ methodologies makes it possible to solve the conflicts in product design without unsystematic trial and error.

There are some interesting studies on applications of TRIZ to service design. Two of the studies are reviewed, and their effectiveness is analyzed in this chapter. One is a study in which the effective use of TRIZ tools is applied to resolve the conflicts in service design [11]. In this research, some TRIZ tools (e.g., Problem Formulator [12], 40 inventive principles [9], and Separation Principles [9]) make up the entire conflict-solving process. The

conflict resolution process is very useful in that a simple flowchart of the conflict resolution process with TRIZ tools is shown. In this study, however, service modeling is ambiguous and largely dependent on each designer. In this sense, success and failure of the entire conflict resolution process are strongly influenced by those of service modeling.

The other is a study in which the modification of 40 inventive principles of TRIZ for service design is suggested [13]. In this research, new principles are proposed by matching the existing principles, and the effectiveness of the two contradictory methods is evaluated. For example, the principle 'Combination-Separation' is described as 'Combine parts of an object or the phases of a process to form a uniform object or process. Separate a uniform object or a uniform process to form independent parts or phases.' However, in this study, the lack of defined service modeling makes it difficult to solve conflicts. In addition, the effectiveness of the new principles remains to be verified, whereas the principles originally proposed in TRIZ are based on the analysis of patents.

Therefore, we believe that the operation of conflict resolution must be based on a defined method of service modeling. Unlike the existing studies reviewed in this section, this paper provides a consistent methodology from service modeling to conflict resolution. In the following sections, the definition and modeling of service in SE are presented. The model of service is a target of the operations of conflict resolution proposed in this paper.

3 SERVICE MODELS IN SERVICE ENGINEERING

3.1 Service Engineering

A new engineering paradigm that aims at reducing the production and consumption volumes of artifacts to an adequate, manageable size while maintaining sustainable economic growth is proposed. This paradigm is called the Post Mass Production Paradigm (PMPP) [14]. Service Engineering (SE) is proposed as one of the solutions for the PMPP. The basic concept of SE was proposed as one of the solutions for PMPP by T. Tomiyama in 2001 [6]. SE aims to provide an engineering methodology for the representation, design, and evaluation of service.

3.2 Definition of service and its components

In SE, a service is defined as 'An activity by a service provider to change the state of a service receiver' [15]. A service is delivered by means of service contents and service channels. While service contents directly change the receiver's state, service channels transfer, amplify, and control service contents and indirectly influence the state change of a service receiver. Service contents and channels constitute the realization structure of a service. Therefore, designers need to reinforce the service contents and channels in order to design a competitive service.

Traditionally, service has mainly been discussed in the field of marketing, and not in engineering. Within marketing, a service is generally defined as a human activity. For example, J.M. Rathmell defined a service as 'A service is a deed, a performance, an effort.' The definition in SE is different from that in marketing in that any activity that causes a positive state change of a customer is regarded as a service. In SE, manufacturing companies are generally considered to provide service contents (e.g., convenience) through products as a service channel. The broadly defined service definition makes it possible to treat products and human activities in a unified framework.

In SE, it is assumed that a service and the state change of a service receiver can be expressed as a combination of parameters that represent the service and the relationships among the parameters [8]. A receiver's state is represented by a set of Receiver State Parameters (RSPs). Contents and channels are expressed by Contents Parameters (CoPs) and Channel Parameters (ChPs), respectively.

3.3 Sub-models of a service: View model

Various sub-models of a service are proposed in SE. However, only one sub-model, called a view model, is explained here. In this paper, a view model is treated as a target of conflict detection.

In order to express a service, it is essential to describe what contents and channels are provided, and how they are provided for a service receiver. In other words, a specific manner of changing a receiver's state is described in a view model. A view model describes a functional structure to realize a change in an RSP and expresses a part of the realization structure of a service through the relationship between the RSP and the functional structure, described in the form of the functional relations among the RSP, contents papameter, and channel parameter [8].

The functions of channels and contents are expressed by function names as lexical expressions and Function Parameters (FPs) as target parameters of functions. Each function is related to other functions. The FPs that are directly related to RSPs are recognized as contents parameters, and those indirectly influencing RSPs are channel parameters. The lowest functions, which have been sufficiently deployed, are related to entities. An entity is something that exists in the real world. An entity includes a product as well as a person, organization, and software. An entity has one or more attribute parameters (APs).

3.4 Describing process of a view model

The describing process of a view model is described below. Figure 1 shows a flow-chart of describing a view model. The numbers correspond to those shown in Figure 1. Figure 2 shows an example of a view model that describes a part of a realization structure of a coffee shop

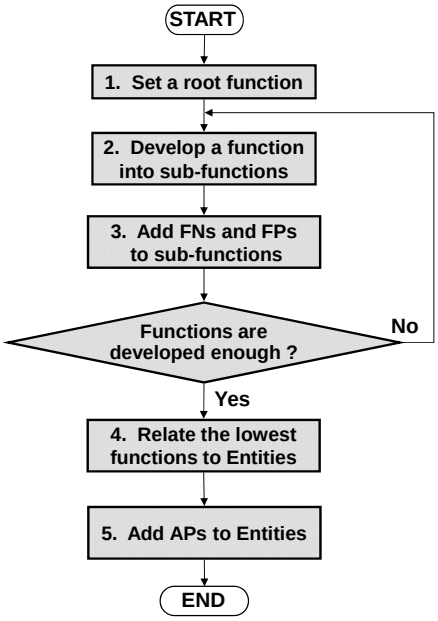

Figure 1: A flow chart of describing a view model.

service.

1. Set a root function.

A root function, which has an RSP as a target parameter, is set. This function is the most abstract function that can cause a state change in the RSP. This function corresponds to a function requirement in product design. For example, a root function ‘make surroundings comfortable’ that has an RSP ‘comfortable surroundings’ is shown in Figure 2.

2. Deploy a function into sub-functions.

A function is deployed into sub-functions and a tree structure is created. When function 1 (F1) is deployed into, function 2 (F2) and function 3 (F3), F1 is fulfilled by F2 and F3. In other words, F1 is influenced by F2 and F3. For instance, the root function ‘make surroundings comfortable’ is deployed into two more detailed functions, ‘control surrounding noise’ and ‘prepare places for customer’ in Figure 2. A seamless deployment from a root function to the lowest functions prevents the omission of functions required to realize a service.

3. Add function names (FNs) and function parameters (FPs)**.**

A function name and a function parameter are added to each function, e.g., ‘control surrounding noise’ as a function name, and ‘volume of surrounding noise’ as a function parameter.

4. Relate the lowest functions to entities.

An entity is related to each lowest function. The entities required to realize a service are described without omission through this process. For instance, a lowest function ‘prepare chairs’ is related to an entity ‘chair’ in Figure 2.

5. Add attribute parameters to entities

Finally, attribute parameters are added to each entity. The required properties of entities are described through this process. In Figure 2, for example, the entity ‘chair’ has three APs, ‘height,’ ‘material,’ and ‘size of seating surface.’

The description of a view model is completed through all these processes. For each RSP, a view model is described in a service.

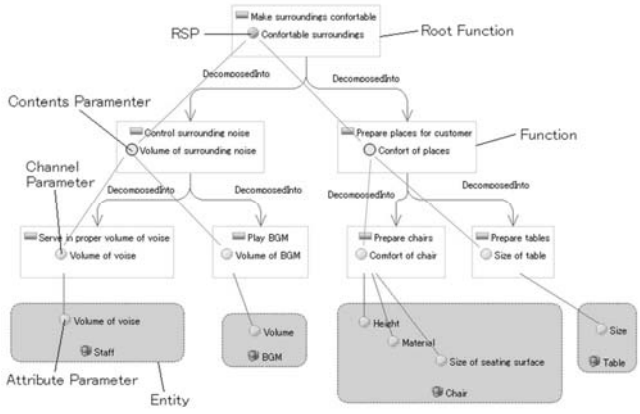

Figure 2: View model of a coffee shop service.

4 DEFINITION OF CONFLICT

In order to develop an argument on conflict, the word ‘conflict’ needs to be clearly defined. In this paper, a conflict is defined as a state, where a designer improves a part of a service design solution and another part of it deteriorates. In the following two chapters, two methods to detect and solve conflicts are explained.

5 AN APPROACH TO CONFLICT RESOLUTION USING LEXICAL EXPRESSIONS OF FUNCTIONS

5.1 Conflict detection with lexical expressions of functions

This is one of the conflict-detection methods. In this method, conflicts in view models are detected using the lexical expressions of functions in view models. In general, a lexical expression of a function consists of an object and a predicate. Therefore, conflict detection is executed, analyzing these elements from the lexical point of view. For example, two functions ‘increase staff’ and ‘decrease staff’ have the antonymous predicates of each other, whereas these functions have the same objects. In general, it is impossible to increase and decrease the staff at the same time; thus, conflict between these two functions is detected. In another case, two functions ‘increase staff’ and ‘decrease employees’ have the antonymous predicates, whereas these functions have the synonymous objects. The conflict between these two functions is also detected in this method.

```
⋮
<Predicate rdf:ID="decrease">
  <IsSynonymOf>
    <Predicate rdf:ID="diminish">
      <IsSynonymOf rdf:resource="#decrease"/>
      <IsSynonymOf>
        <Predicate rdf:ID="lessen">
          <IsAntonymOf>
            <Predicate rdf:ID="multiply">
              <IsSynonymOf>
                <Predicate rdf:ID="enlarge">
                  <IsAntonymOf rdf:resource="#decrease"/>
                  <IsSynonymOf rdf:resource="#increase"/>
                </Predicate>
              </IsSynonymOf>
              <IsAntonymOf rdf:resource="#decrease"/>
              <IsAntonymOf rdf:resource="#lessen"/>
            </Predicate>
          </IsAntonymOf>
          <IsSynonymOf rdf:resource="#diminish"/>
        </Predicate>
      </IsSynonymOf>
      <IsSynonymOf>
        <Predicate rdf:ID="minifyf">
          <IsSynonymOf rdf:resource="#diminish"/>
        </Predicate>
      </IsSynonymOf>
    </Predicate>
  </IsSynonymOf>
  <IsAntonymOf rdf:resource="#enlarge"/>
  <IsAntonymOf rdf:resource="#increase"/>
</Predicate>
⋮
```

Figure 3: A screen shot of a Lexical Relation Database.

5.2 Building up Lexical Relation Database

In order to detect conflicts using the above method, frequently used terms representing synonymous and antonymous relations, need to be accumulated and kept ready for searching. Therefore, a database (Lexical Relation Database) including such information needs to be created on a computer. However, it is almost impossible to accumulate all these terms in it, since there exist a very large number of terms representing such relations. Therefore, the database should be expandable and offer the option to designers or system administrators to add new terms. The number of terms in the database that can be considered to be adequate largely depends on an individual designer. However, it is considered to be sufficient for a default database to store a number of terms equal to that included in a commercial thesaurus.

The requirements given above for a Lexical Relation Database are developed using an RDF [16] and an OWL [17], which are used extensively in the research field of Semantic Web. A property [16] makes it possible to describe the terms and relationships among them. In this detection method, two properties are additionally defined by the authors: 'IsSynonymOf' and 'IsAntonymOf' for the synonymous and antonymous relationships, respectively. 'IsSynonymOf' is defined to be a transitive property [17]. Thanks to this definition, for example, when a pair ('staff' and 'employee') is an instance of IsSynonymOf and another pair ('employee' and 'worker') is an instance of IsSynonymOf, a pair ('staff' and 'worker') is also assumed to be an instance of IsSynonymOf from the existing information in the database. Figure 3 shows a screen-shot of a Lexical Relation Database that was created by the authors made as a prototype.

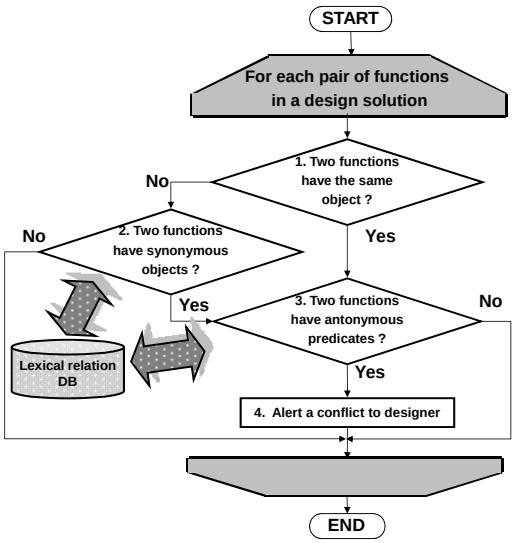

Figure 4: Flowchart of Conflict Detection Using Lexical Expressions of Functions.

5.3 Process of conflict detection

Figure 4 is the flowchart of conflict detection using the above-mentioned Lexical Relation Database. The details of the conflict detection process are explained below.

The lexical expressions of the functions in the design solution (or a part of it) are compared by pairwise comparison. For each pair of functions, the following process is executed. The numbers correspond to those in Figure 4.

1. Determine whether the two functions have the same object.

It is determined whether or not the two functions have the same object. For instance, two functions 'increase staff' and 'decrease staff,' share the object 'staff.'

2. Determine whether the two functions have the object in a synonymous relation.

When the two functions do not have the same object in common, it is determined whether or not these functions have the object in a synonymous relation by accessing the Lexical Relation Database.

3. Determine whether the two functions have the predicate in an antonymous relation.

When the two functions share the same or synonymous objects, it is determined whether or not these functions have antonymous predicates by accessing the Lexical Relation Database.

4. Alert a designer to conflict.

When a conflict is detected in the above processes, the designer is alerted to the conflict.

5.4 Conflict-solving with TRIZ tool

To solve a conflict detected with the above method, Separation Principles in TRIZ are useful. Separation Principles are the ways a designer can solve a conflict within a parameter itself (physical conflict [8]) in TRIZ. When two different requirements are demanded for a parameter, the parameter can be separated in the view point of time or place with these principles. Separation in the view point of time, for example, provides a solution: the parameter is set to fulfill the requirement A in a certain period of time, and is set to fulfill the requirement B in another period of time.

The two functions with a conflict detected in the above-mentioned method have the same (or synonymous) object and antonymous predicate. One function requires the object to fulfill the requirement A, and the other requires it to fulfill the requirement B. A predicate behaves

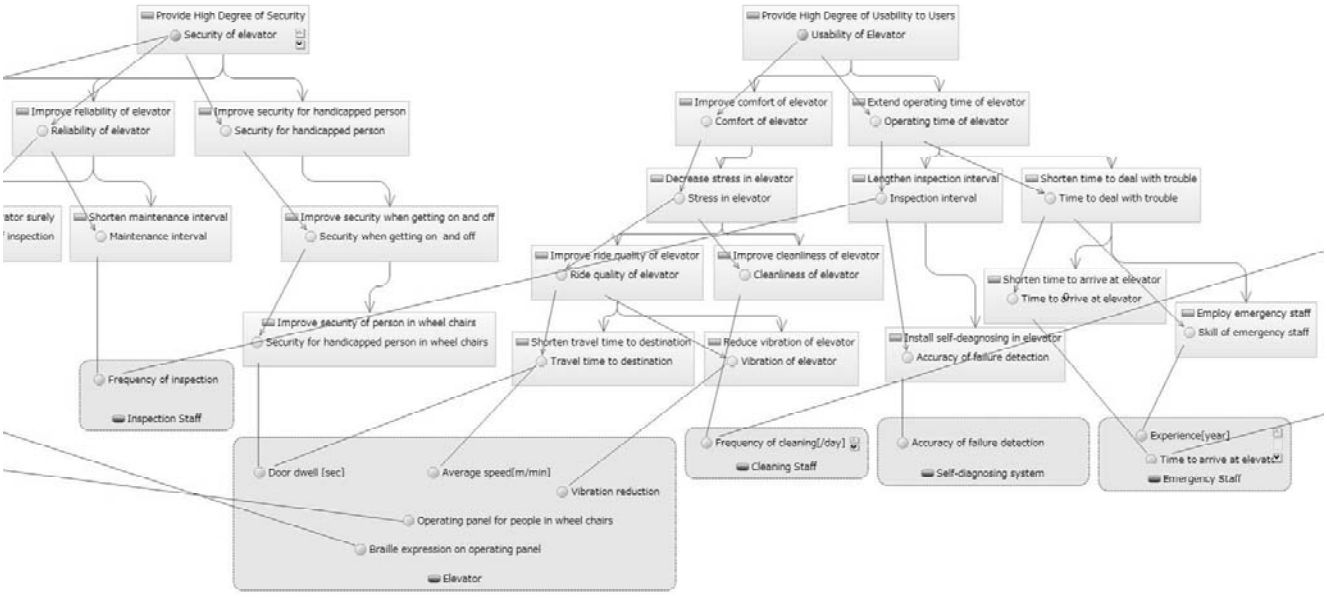

Figure 5: View models for elevator installation service.

as an operant that requires an object to fulfill a requirement. Therefore, the demands required by these two functions cannot generally be fulfilled at the same time. The use of Separation Principles, however, makes it possible that an object fulfills the requirement A at one time and the requirement B at another time.

5.5 Verification experiment using the proposed method

The method proposed above is verified by applying it to an existing service case. The methods are applied to view models of an elevator company's business. This type of business is one of the most 'servicized' businesses in that it regards an installation, preventive maintenance, repairs, and remodeling as profit centers. The authors described the view models of the elevator installation and maintenance service as a target of conflict detection (Figure 5). The process of conflict detection is explained below.

In the view models, a function 'lengthen inspection interval' is required to fulfill the upper function 'extend operating time of elevator,' while another function 'shorten maintenance interval' is required to fulfill the upper function 'improve reliability of elevator.' The process of conflict detection between the function 'lengthen inspection interval' and the function 'shorten maintenance interval' is as follows.

First, it is checked whether those two functions have the same object. However, it is found that they do not have the same object. Therefore, it is checked whether or not the two functions have the objects in a synonymous relation by accessing the Lexical Relation Database. As a result, it is determined that these functions have synonymous objects.

Second, it is checked whether or not 'lengthen inspection interval' and 'shorten maintenance interval' have predicates in an antonymous relation by accessing the Lexical Relation Database. Consequently, it is determined that these two functions have antonymous predicates. As a result of the above process, a conflict between these two functions is successfully detected; they simultaneously have synonymous objects and antonymous predicates.

The Separation Principles described in TRIZ are applied as the next step in the solution. In this case, a separation in time is helpful to solve the problem. For example, when an office building (not a personal residence) is a client of the service, a given time can be divided into 'during business hours' and 'outside business hours.' The following can be a solution: inspect the elevator outside business hours, except for the maintenance menus that should be performed during business hours, and inspect the elevator frequently outside business hours.

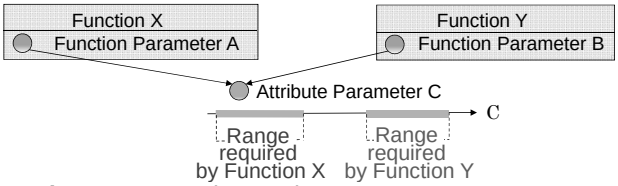

Figure 6: Inconsistency between parameter ranges.

6 AN APPROACH TO CONFLICT RESOLUTION USING RANGES FOR DESIGN PARAMETERS

6.1 Conflict detection with the ranges of design parameters

Presented here is another method of detecting conflict. In this method, the conflicts between functions in view models are detected by using the ranges of design parameters, while the lexical expressions are utilized in the method proposed in the previous chapter. Take a part of a view model that a parameter influences multiple upper parameters of the functions for example, as shown in Figure 6. The information of the ranges of parameter C is used in order to detect a conflict between function X and function Y. If there is no shared value between these two ranges, the designer is alerted to the conflict.

The state with no shared value between two ranges suggests that the design solution requires infinitely large information to implement it, from the perspective of 'The Information Axiom' of N. P. Suh's Axiomatic Design [18]. This state means that the service is unfeasible without reconsideration of service design.

6.2 Range calculation with Set Based Theory

The range required to develop the upper functions must be calculated for each parameter that influences multiple functions. Thus, the Set Based Theory [19], which is a theory that formulates parametric operations with a range, is utilized. In the Set Based Theory, the variation of each

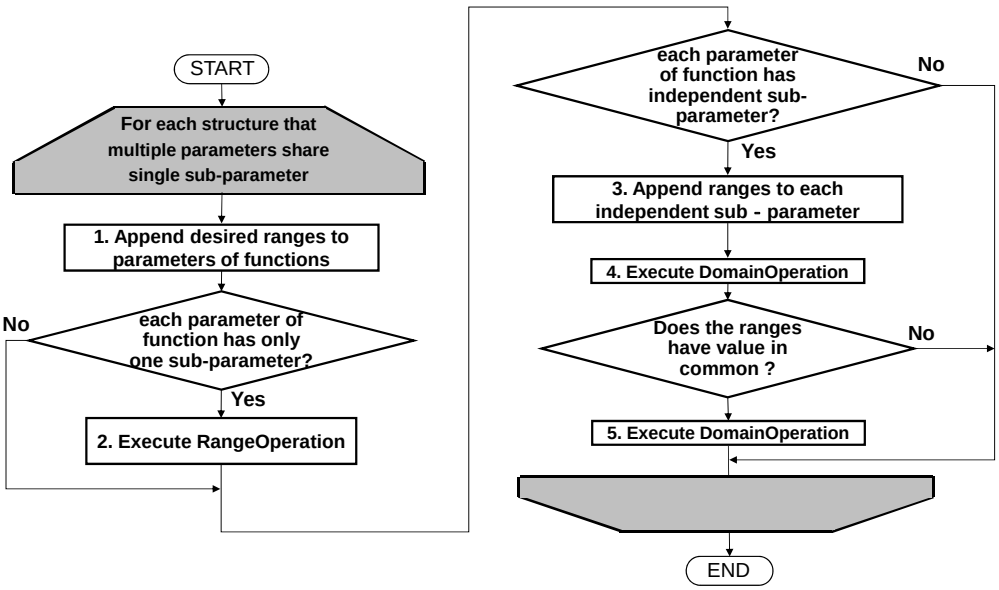

Figure 7: Flowchart of conflict detection using values of design parameters.

variable in a system is expressed as a set. In addition, the set is expressed as an interval. The influence of each variation within a parameter on any other parameter is calculated with the rule of interval operation. The Set Based Theory consists of four operations; however, only RangeOperation and DomainOperation are used in this method. The RangeOperation is an operation that the range of an unknown dependent value is calculated from the ranges of multiple independent valuables, and the DomainOperation is an operation that the range of an unknown independent valuable is calculated from a dependent valuable and multiple independent valuables. The Set Based Theory is introduced in some studies on product design methodology (e.g., [20]).

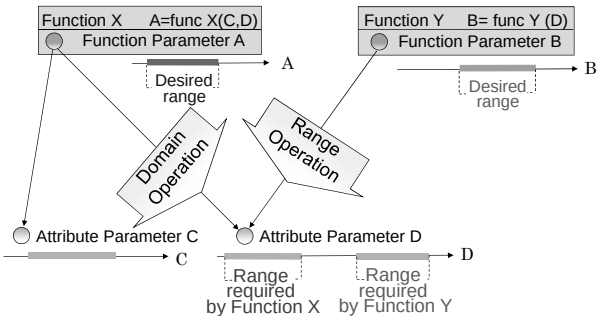

Figure 8: Detection of inconsistency between ranges of the parameters.

6.3 Process of conflict detection with Set Based Theory

Figure 7 is a flowchart of conflict-detection using the Set Based Theory described above. The details of the conflict-detection process are described in this section with an example of a part of a view model (Figure.8), where a parameter influences on multiple upper parameters of the functions. For each structure, the conflict-detection method is executed through the process outlined below. The numbers correspond to those in Figure 7.

1. Append the desired ranges to parameters of functions.

The desired ranges to parameters of functions, parameters A and B, are appended.

2. Execute RangeOperation.

The range of parameter D, which influences multiple upper functions, is calculated with Set Based Theory. In this case, the value of parameter B is determined by only parameter D. Therefore, the range of parameter D, which is required to put parameter B within the desired range, is calculated by using RangeOperation.

3. Append ranges to each independent sub-parameter.

In this case, RangeOperation cannot be utilized to calculate the ranges of parameter D, because the value of parameter A is determined by parameters C and D. Thus, the range of independent parameter C is preliminarily determined.

4. Execute DomainOperation.

DomainOperation is executed in order to calculate the range of parameter D required to put parameter A within the desired range.

5. Alert the designer to conflict.

In process 1 to 4, two different ranges for parameter D are calculated. One is the range required to put parameter A within the desired range, and the other is the range required to put parameter B within the desired range. If these two ranges do not have value in common, the designer is alerted to the mutual conflict between function X and Y.

6.4 Conflict-solving with TRIZ tool

In order to solve a conflict detected with the above method, there are two possible solutions. One is almost identical to the one used in the previous chapter. In this method, a detected conflict is solved using Separating Principles. Separation in time, for instance, enables for value of a parameter to be X_1 at a time, and X_2 at another time. The other solution is the method with a Technical Contradiction Table in TRIZ methodology. Details of this method are provided below with Figure 8 again.

1. Determine the parameters influenced by a parameter.

Parameter D is determined to be conflicting using the Set Based Theory; it is required to be in mutually exclusive ranges simultaneously. Parameter A and B, which are influenced by parameter D, are utilized in this conflict resolution method.

2. Apply each parameter to one of the 39 attributes.

In the Technical Contradiction Table, a list of 39 attributes is defined, such as size, weight, and ease of manufacture. The Technical Contradiction Table is a 39x39 grid matching each attribute to another. In each cell, there is a list of inventive principles. Therefore, in order to use this table, each conflicting parameter is applied to one of the 39 attributes.

3. Use of the Technical Contradiction Table.

Possible inventive principles are identified by matching one parameter to another. If there is a useful principle in a cell, it is utilized to solve the conflict.

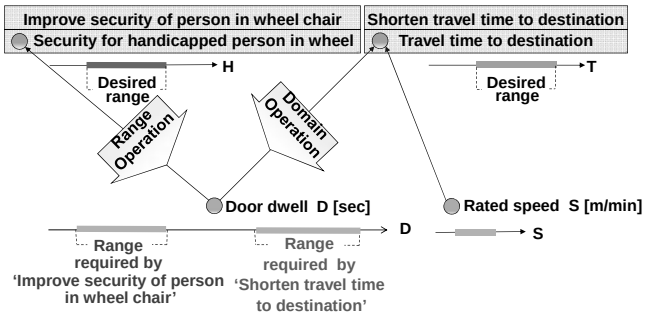

Figure 9: Detection of inconsistency between parameter ranges 'Security for handicapped person in wheel chair' and 'travel time to destination.'

6.5 Verification experiment of the proposed method

In this section, a verification experiment for the method proposed in this chapter is conducted. The method is applied to the view models of the elevator installation and maintenance service. Figure 9 shows a part of the structure extracted from the view models of the service (Figure 5). Figure 9 shows two functions, 'improve security of handicapped person in wheel chair' and 'shorten travel time to destination.' It is assumed that the parameter, 'security of handicapped person in wheel chair' is formulated as formula (1), where H and D represent 'security for handicapped person in wheel chair' and 'door dwell,' respectively. Formula (1) gives H = 0.5 when D = 10 [sec], and ranges from 0 (very dangerous) to 1 (very safe). 'Security for handicapped person in wheel chair' is quantified with 'door dwell' in a simple and easy way.

$$H = \frac{1}{1 + e^{0.25(10-T)}} \quad (1)$$

Further, it is assumed that the parameter, 'travel time to destination' is formulated as formula (2), where T and S

represent ‘travel time to destination’ and ‘rated speed,’ respectively. Formula (2) gives T = 0.5 when D = 7 and S = 80 [m/min], and ranges from 0 (very slow) to 1 (very fast). ‘Travel time to destination’ is quantified with ‘rated speed’ and ‘door dwell’ in a simple and easy way.

$$T = \frac{0.5}{1+e^{0.25(80-S)}} + \frac{0.5}{1+e^{0.5(D-7)}} \quad (2)$$

The process of conflict detection is as follows. The desired ranges to the parameters of the functions ‘improve security of handicapped person in wheel chair’ and ‘shorten travel time to destination’ are described as H = [0.7, 0.9] and T = [0.6, 0.9], respectively. The range of D required by ‘improve security of person in wheel chair’ is calculated as D_H = [13.4, 18.8] with RangeOperation. In addition, the range required by ‘shorten travel time to destination’ is calculated as D_T = [3.1, 9.8] with DomainOperation, after the range of independent parameter S is determined as S = [90, 120]. As a result, D_H and D_T have no values in common. Therefore, it is determined that there is a conflict between them.

In this case, the detected conflict can be solved using the Technical Contradiction Table in TRIZ. Two parameters, ‘security for handicapped person in wheel chair’ and ‘travel time to destination’ are applied to ‘harmful side effects’ and ‘waste of time,’ respectively. As a result, the principle ‘Segmentation’ is found as a way to solve the conflict. ‘Segmentation’ gives designers advice to divide an object into independent parts or fragmenting parts. This principle suggests that two different operating panels of elevator make it possible to divide the behavior of the door. When the panel for healthy people is used, the door should close faster, and when the panel for handicapped people is used, the door should close more slowly. In fact, this idea is used in many elevators available in the market. The authors are not sure that the elevators are designed using the TRIZ methodology. However, the usefulness of the solution obtained by using this method is shown.

7 DISCUSSION

7.1 Effectiveness of the proposed methods

Two different methods to detect and solve conflicts are proposed. The two approaches have allowed for an expanded conflict resolution.

Conflict resolution with lexical expressions of functions

The use of conflict detection using lexical expressions of functions allows designers to grasp the possiblity that the design solution includes conflicting description, which could be a serious obstacle in providing the service. This method has great effectiveness especially in earlier design stage, since it is rare that the design information contains a good deal of specific data (i.e., numerical data). Earlier detection of potential conflicts enables designers to consider design change earlier, which could consume additional time and money. The method proposed in this papaer intends to detect conflicts in the design solution described in the form of view model (function structure). However, it can cover design documents in other formats, combining with natural language processing technology. Therefore, the idea of scanning design documents for conflicting descriptions by using automated system would be greatly helpful also in industry.

Conflict resolution with ranges of design parameters

The use of conflict detection using ranges of design parameters allows designers to find the possibility that the design solution includes numerically conflicting description. This method enables a conflict resolution in later design phase, where detailed specification of service activity and product, i.e., number of staff or timing in service delivery process or values of parameters in product used in service delivery. In many cases, designers face a dilemma whether it is obvious or unobvious one; a parameter has an impact on several functions and it has to have mutually exclusive values for each functions. The method proposed in this paper allows designers to comprehend ‘which parameter is a conflicting factor and which functions are related with it.’

7.2 Possible improvements of the proposed methods

The following possible improvements for the proposed methods have been identified:

Limitation of the function expression that the method can cover

In conflict resolution with lexical expressions, the form of function expression is limited and functions not following the form are not to be targets of detection. That is, the proposed method can be used to detect the conflicts only from a realization structure, in which functions are expressed by a predicate and an object. Usually, functions are expressed in various forms. For instance, functions can be expressed with a subject, modifiers, and a direction. Therefore, the detection method should be modified to cover such functions.

Improvement of detection accuracy, introducing an external dictionary

The detection method with lexcal expressions can search for conflicts only from the Lexical Relation Database built by designers or an administrator of the detection system. The database has the advantage of being expandable if necessary to appropriately reflect the users’ needs or the feature of the design objects. However, it is likely that the conflicts detected by using the database are obvious and designers could detect them (if they didn’t mind taking time and trouble). To solve this issue, the authors are planning to introduce an external thesaurus dictionary as an additional database. The additional database would allow designers to catch not only more ‘obvious’ conflicts but also more ‘unobvious’ ones, which they hadn’t expected.

The way of formulating qualitative parameters

In the previous chapter, the realization of an elevator installation and maintenance service is used as an example in verification experiment. The formulation of relationships between parameters including highly qualitative ones is essential to practice conflict detection with Set Based Theory. In this paper, qualitative parameters (e.g., ‘security for handicapped person in wheel chair’) are assumed to range from 0 to 1. However, the validity of this is left to be reconsidered. In general, human activities such as ‘politeness’ and ‘motivation’ are involved in services. Therefore, the method of treating qualitative parameters is a big issue.

The need for the extended TRIZ tool

As for the method to solve the conflict detected with the ranges of design parameters, the Technical Contradiction Table in TRIZ is utilized. However, this tool was originally developed on the assumption that it is to be used for product design. Most of the attributes listed in the Technical Contradiction Table are engineering parameters. Therefore, the modification of the Technical Contradiction Table for service design makes the method more useful.

Implementation in Service CAD

The authors will implement the methods proposed in this paper as a computer system in Service CAD.

8 CONCLUSION

In this paper, a series of methods to detect and solve conflicts is proposed. The effectiveness of the proposed methods has been proven through the verification experiments.

9 ACKNOWLEDGEMENTS

This research was partially supported by the Ministry of Education, Science, Sports, and Culture through a Grant-in-Aid for Scientific Research (B), 18360079, 2006.

10 ACKNOWLEDGEMENTS

[1] Shostack, G.L., 1981, How to Design a Service, in Donnelly, J.H. and W.R. George, eds., Marketing of Services, American Marketing Association.

[2] Aurich, C., Fuchs, C., and DeVries, F., 2004, An Approach to Life Cycle Oriented Technical Service Design, Annals of the CIRP, 53-1, pp. 151-154.

[3] Gupta, S. K., Hayes, C. C., Ishii, K., Kazmer, D., Sandborn, P. A., and Wood, W. H., 2004, New Directions in Design for Manufacturing, Proceedings of ASME 2004 Design Engineering Technical Conferences and Computers and Information in Engineering Conference -DETC04-, CD-ROM.

[4] Donaldson, K. M., Ishii, K., and Sheppard, S. D., 2004, Customer Value Chain Analysis, Proceedings of ASME 2004 Design Engineering Technical Conferences and Computers and Information in Engineering Conference -DETC04-, CD-ROM.

[5] Chesbrough, H., and Spohrer, J., 2006, A Research Manifesto for Services Science, Commun. ACM 49, 7 (Jul. 2006), 35-40.

[6] Tomiyama, T., 2001, Service Engineering to Intensify Service Contents in Product Life Cycles, Proceedings of the Second International Symposium on Environmentally Conscious Design and Inverse Manufacturing (Eco Design 2001), IEEE Computer Society, pp.613-618.

[7] Shimomura, Y., and Tomiyama, T., 2002, Service Modeling for Service Engineering, Proceedings of the 5th International Conference on Design of Information Infrastructure Systems for Manufacturing 2002 -DIISM2002-, pp. 309-316.

[8] Shimomura, Y., Sakao, T., Sundin, and E., Lindahl, M., 2007, A Design Process Model and a Computer Tool For Service Design, Proceedings of the ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2007, CD-ROM

[9] Mann, D., 2002, Hands On: Systematic Innovation, Creax

[10] Mann, D., Domb, E., 1999, 40 inventive (business) principles with examples, The TRIZ Journal, Sept issue, (http://www.triz-journal.com/archives/1999/09/a/index.htm).

[11] Zhang, J., Tan, K., Chai, K., 2003, Systematic innovation in service design through TRIZ, Proceedings of the EurOMA-POMS 2003 Annual Conference, Vol.1, pp. 1013-1022.

[12] Zlotin, B., Zusman, A., 2001, Directed Evolution: Philosophy, Theory and Practice, Ideation International Inc, Southfield, MI.

[13] Ruchti, Bruno., Livotov, P., 2001, TRIZ-based Innovation Principles and a Process for Problem Solving in Bussiness and Management, Proceedings of the European TRIZ Association

[14] Tomiyama, T., Sakao, T., Umeda, Y., Baba., 1995, The Post Mass Production Paradigm, Knowledge Intensive Engineering, and Soft Machines, Proceedings of Life Cycle Modeling for Innovative Products and Processes, Chapman & Hall, London, pp.369-380.

[15] Shimomura, Y., Watanabe, K., Arai, T., Sakao, T., and Tomiyama, T., 2003, A Proposal for Service Modeling, Proceedings of the 3rd International Symposium on Environmentally Conscious Design and Inverse Manufacturing (EcoDesign 2003), pp. 75-80.

[16] Beckett, D., 2004, RDF/XML Syntax Specification (Revised), (http://www.w3.org/TR/rdf-syntax-grammar/).

[17] Patel-Schneider, P., Hayes, P., Horrocks, I., OWL Web Ontology Language Semantics and Abstract Syntax, (http://www.w3.org/TR/owl-semantics).

[18] SUH, N. P., 1990, THE PRINCIPLES OF DESIGN, OXFORD UNIVERSITY PRESS.

[19] Finch, W. W., Ward, A. C., 1995, Generalized set-propagation operations over relations of more than three variables, Artificial Intelligence for Engineering Design, Analysis and Manufacturing 1995, pp. 231-242.

[20] Umemori, Y., Kondoh, S., Umeda, Y., Shimomura, Y., Yoshioka, M., 2001, Design For Upgradable Products Considering Future Uncertainty, In Proceedings of Eco Design 2001, IEEE, pp. 87-92.